AF
JFW

Docket No.: 0826.1718

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re the Application of:

Yoshitake SHINKAI

Serial No. 09/817,288 Group Art Unit: 2152

Confirmation No. 7742

Filed: March 27, 2001 Examiner: Lesniewski, Victor D.

For: FILE REPLICATION SYSTEM, REPLICATION CONTROL METHOD, AND STORAGE MEDIUM

APPELLANTS' BRIEF IN REPLY UNDER 37 C.F.R. § 41.41

Mail Stop Appeal Brief-Patents
Commissioner for Patents
PO Box 1450
Alexandria, VA 22313-1450

Dear Sir:

In response to the Examiner's Answer mailed August 3, 2007 in the above-identified application, Appellants hereby submit the present Reply Brief.

RESPONSE TO ARGUMENTS

1. Responses to Arguments that the subject matter of claims 1-3, 8, 10-12, 14, 23, 25, 27, and 28 are unpatentable over U.S. Patent No. 5,964,886 (Slaughter) in view of U.S. Patent No. 5,634,122 (Loucks).

Claims 1 and 10

> On page 4 of the Examiner's Answer, the Examiner alleged:
>
> *Concerning the independent claims, Slaughter did not explicitly state a token manager that gives access permission for a file when no other node has update permission. . ..* However, Loucks remedies the token manager issue as his system (also focused on accessing a file on a storage device) utilizes a manager to grant or deny tokens to requesting network nodes. Similarly, since Slaughter does not deal explicitly with tokens, *Slaughter also does not state notifying the requesting node of another node that has a token.* However, *Slaughter does operate similarly in that his system maintains a list of active devices and when needed informs the questing node of an alternate node through which the requested file can be accessed.*

See Examiner's Answer, page 4, lines 4-13 [emphasis added].

Appellants respectfully submit that Loucks does not remedy the token manager deficit of Slaughter. In particular, in contrast to the present invention's token manager, the token manager of Loucks does not disclose or suggest "otherwise issuing a notification of a permitted node that has update permission for the file in response to an access request in the first node," as recited in claim 1, for example. Rather, Loucks specifically states that if the token has been previously granted to a first process, the "ltkm" revokes the token from the first process before granting the token to the new requesting process. *See* Loucks, column 7, lines 7-12. Therefore, Applicants respectfully submit that Loucks teaches away from the present invention in that instead of notifying the node of a process that currently has the token, Loucks revokes the token from the process currently holding the token and provides the new requesting process with the token.

Appellants further respectfully submit that the citations to Slaughter by the Examiner (for example, column 10, lines 16-28) merely reference how data access *requests* are processed, not what occurs in response to the data access requests. Therefore, the Examiner has failed to provide specific evidence in Slaughter of the token manager function of the present invention. Moreover, the Examiner specifically stated that "Slaughter did not explicitly state a

token manager that gives access permission for a file when no other node has update permission."

In response to Appellants' argument regarding the combination of Slaughter and Loucks failing to disclose the claimed feature directed to an IO Request Intercepting Portion accepting access to a file, the Examiner alleged that Slaughter's functionality regarding routing a data access request to a primary node meets the claimed feature.

Appellants respectfully submit that regardless of whether access occurs in the IO Request Intercepting Portion or in the first node ("as a whole") in the present invention, Slaughter does not disclose or suggest the claimed feature.

In particular, as illustrated in FIG. 3 of Slaughter, the netdisk drive 318A is included within a node 104A. As the netdisk drive 318A forwards the data access request to another node, access does not occur in the netdisk drive 318A. Moreover, Slaughter provides no information indicating that the primary node to which the netdisk drive 318A forwards the data access request is a part of the same node in which the netdisk drive is included. In fact, the primary node and the node including the netdisk drive 318A appear to be completely separate nodes.

In response to Appellants' argument regarding the combination of references failing to disclose or suggest the claimed feature directed to "asking the permitted node that has update permission for the file to access the file," the Examiner alleged that Slaughter's primary node handles the data access request "when active (meaning access request taking place in the first node)." Appellants respectfully submit that the Examiner's argument is not supported by the reference. In particular, for example, the reference fails to define the term "active." In fact, the reference suggests that the term "active" simply refers to whether a node is operational, regardless of whether the node has access permission, as one of the goals of the reference is performing access requests in the presence of "failure" or inactivity of nodes. *See* Slaughter, column 2, lines 17-23.

Moreover, in contrast to the present invention in which the node not having access permission asks another node having access permission for data access, the primary node of Slaughter does not ask the secondary node for access permission. Rather, the netdisk drive 318A makes the decision as to which of the nodes (that is, primary node or secondary node) will receive the data access request.

Claims 2, 8, 11, and 12

Appellants respectfully submit the same comments for claims 2, 8, 11, and 12 as were submitted above for claims 1 and 10.

Claims 14, 27, and 28

On page 19 of the Examiner's Answer, the Examiner stated:

> In response to argument 5 (set forth on pages 9-10 of the brief under heading C2), the combination of Slaughter and Loucks does disclose causing an access requesting node to access a file of the access requesting node itself as recited in claim 14. This argument is substantially the same as argument 1 and thus it is maintained that the combination of Slaughter and Loucks teaches these features for the same reasons as given above. See the response to argument 1 above

. Applicants respectfully submit that claim 14, for example, is patentable over the cited combination of references, as neither Slaughter nor Loucks, alone or in combination, discloses or suggests, "causing an access requesting node to access a file of the access requesting node itself when the access requesting node has the latest data of the file and has or is able to obtain access permission from another node having update permission for the file."

Slaughter states that the client 312A sends a data access request to ND 318A. According to Slaughter, the ND 318A then determines to which node to convey the data access request. In particular, the ND 318A routes the data access request to the primary node if the primary node is active or to the secondary node if the primary node is not active. Therefore, in Slaughter, the node requesting data access, that is, the client 312A does not access one of its own files. Rather, it sends a request to the ND 318A, which determines which node will perform the data access.

Conclusion

Applicants respectfully submit that the Examiner has not established a *prima facie* case of obviousness by a preponderance of the evidence for the relevant claims. Reversal of the rejection is, therefore, requested.

Respectfully submitted,

STAAS & HALSEY LLP

Date: 10/1/07

By: _____
Reginald D. Lucas
Registration No. 46,883

1201 New York Avenue, NW, 7th Floor
Washington, D.C. 20005
Telephone: (202) 434-1500
Facsimile: (202) 434-1501

Claims Appendix

1. (previously presented) A file replication system having a plurality of nodes connected to a network, files being distributed to the nodes, wherein

a first node of the nodes comprises:

a first token managing portion giving access permission for a file within the first node when no other node has update permission and otherwise issuing a notification of a permitted node that has update permission for the file in response to an access request in the first node, and

an IO request intercepting portion accepting an access to the file, the access taking place in the first node when said IO request intercepting portion is capable of acquiring the access permission, asking said first token managing portion to acquire the access permission against the access request, and asking the permitted node that has update permission for the file to access to the file when said first token managing portion is not capable of acquiring the access permission, and

a second node comprises a second token managing portion notifying a requesting node that requests the access permission for the file of the permitted node that has the update permission for the file as a response message.

2. (previously presented) A node, connected to at least one other node through a network, every node having a copy of files synchronized with files of other nodes for high availability and high performance to provide a replicated file system, comprising:

a token managing portion managing an access request for a file; and

an IO request intercepting portion asking said token managing portion to acquire access permission for the file against an access request to the file in said node, said token managing portion giving access permission when no other node has update permission for the file and said token managing portion notifying said IO request intercepting portion of another node that has the update permission when the other node has the update permission for the file, in response to the access request of said IO request intercepting portion, said IO request intercepting portion accessing the file in said node when the IO request intercepting portion is capable of acquiring the access permission and said IO request intercepting portion asking the other node that has the update permission to access the file instead of accessing the file in said node when said IO request intercepting portion is not capable of acquiring the access permission.

3. (previously presented) The node according to claim 2, further comprising:

a system structure managing portion performing a restoration process of data of a file of the node when it is newly joined to a system,

wherein while said system structure managing portion is restoring the file, when an access request for the file takes place in the node, said IO request intercepting portion asks another node that shares the file to access the file.

4. (previously presented) The node according to claim 2, further comprising:

a changed data notifying portion propagating an updated content of the file to other node along with information that represents a dependent relationship with another update; and

a received data processing portion reflecting the updated content to the file while assuring an order of the update based on the dependency relationship.

5. (previously presented) The node according to claim 4, further comprising:

a system state information portion storing information about propagation mode of an updated content for each of at least one file,

wherein said changed data notifying portion propagates the update content based on information queued in said system information portion.

6. (previously presented) The node according to claim 5, wherein the propagation mode is one of a synchronous mode in which it is assured that the updated content is propagated to all the nodes that share the file, a semi-synchronous mode in which it is assured that the updated content is propagated to the majority of nodes that share the file, and an asynchronous mode in which it is not acknowledged that the updated content is propagated to the nodes that share the file.

7. (previously presented) The node according to claim 4, wherein said system state information storing portion keeps information about each node that shares at least one file for each file.

8. (previously presented) A node, connected to at least one other node through a network, every node having a copy of files synchronized with files of other nodes for high availability and high performance to provide a replicated file system, comprising:

a token managing portion asking another node to acquire an access permission for a file against an access request for the file in said node; and

an IO request intercepting portion accepting an access request for a file in said node,

asking said token managing portion to acquire the access permission for the file against the access request to the file in said node, said token managing portion giving access permission when no other node has update permission for the file and otherwise notifying said IO request intercepting portion of another node that has the update permission for the file, said IO request intercepting portion accessing the file in said node when the IO request intercepting portion is capable of acquiring the access permission and asking the other node that has the update permission for the file to access the file according to the access request instead of accessing the file in said node when said token managing portion is not capable of acquiring the access permission for the file.

9. (cancelled)

10. (previously presented) A file replication system having a plurality of nodes connected to a network, files being distributed to the nodes, wherein

a first node of the nodes comprises:

first token managing means for giving access permission for a file within the first node when no other node has update permission and otherwise issuing a notification of a permitted node that has update permission for the file in response to an access request in the first node, and

IO request intercepting means for accepting an access to the file, the access taking place in the first node when said IO request intercepting portion is capable of acquiring the access permission, asking said first token managing means to acquire the access permission against the access request, and asking the permitted node that has update permission for the file to access to the file when said first token managing means is not capable of acquiring the access permission, and

a second node comprises second token managing means for notifying a requesting node that requests the access permission for the file of the permitted node that has the update permission for the file as a response message.

11. (previously presented) A node, connected to at least one other node through a network, every node having a copy of files synchronized with files of other nodes for high availability and high performance to provide a replicated file system, comprising:

token managing means for managing an access request for a file; and

IO request intercepting means for asking said token managing means to acquire an access permission for the file in response to an access request to the file in said node, said

token managing means giving access permission when no other node has update permission for the file and said token managing portion notifying said IO request intercepting means of another node that has the update permission when the other node has the update permission for the file, in response to the access request of said IO request intercepting means, said IO request intercepting portion accessing the file in said node when the IO request intercepting portion is capable of acquiring the access permission and said IO request intercepting means asking the other node that has the update permission to access the file instead of accessing the file in said node when said IO request intercepting means is not capable of acquiring the access permission.

12. (previously presented) A node, connected to at least one other node through a network, every node having a copy of files synchronized with files of other nodes for high availability and high performance to provide a replicated file system, comprising:

token managing means for asking another node to acquire an access permission for a file against an access request for the file in said node; and

IO request intercepting means for accepting an access request for a file in said node, asking said token managing means to acquire the access permission for the file against the access request to the file in said node, said token managing portion giving access permission when no other node has update permission for the file and otherwise notifying said IO request intercepting portion of another node that has the update permission for the file, said IO request intercepting portion accessing the file in said node when the IO request intercepting portion is capable of acquiring the access permission and asking the other node that has the update permission for the file to access the file according to the access request instead of accessing the file in said node when said token managing means is not capable of acquiring the access permission for the file.

13. (cancelled)

14. (previously presented) A file replication control method for a system having a plurality of nodes connected to a network, files being distributed to the nodes, comprising:

causing an access requesting node to access a file of the access requesting node itself when the access requesting node has the latest data of the file and has or is able to obtain access permission from another node having update permission for the file; and

asking the other node to access the file when the other node has the update permission for the file which is given to only one node at a time.

15. (cancelled)

16. (previously presented) The file replication control method according to claim 14, wherein the other node that has the update permission releases the update permission after an update that has a dependent relationship with the update performed at the other node has been propagated to all the nodes.

17. (previously presented) The file replication control method according to claim 14, wherein said method further comprises:

the other node that has updated the file asynchronously propagating an updated content to the other nodes; and

causing the other node that has updated the file to process an access request that takes place in the access requesting node while the updated content is being propagated.

18. (original) The file replication control method according to claim 17, wherein the updated content is reflected in such a manner that order thereof is assured.

19. (original) The file replication control method according to claim 18, wherein a dependency information that represents order of other updates to be propagated to the other node along with the updated content.

20. (previously presented) The file replication control method according to claim 19, wherein a node that has received the updated content to reflect the updated content on a file of the node itself after receiving a previous updated content based on the dependency information.

21. (previously presented) The file replication control method according to claim 14, wherein a propagation mode of an updated content is designated for each of at least one file.

22. (previously presented) The file replication control method according to claim 14, wherein a node to which an updated content is propagated is designated for each of at least one file.

23. (previously presented) The file replication control method according to claim 14, further comprising:

restoring data of a file of a newly joined node; and

operating a user program before data of the file is completely restored.

24. (previously presented) The file replication control method according to claim 23, wherein restored data is transmitted in such a manner that order of update requests for the file is assured.

25. (previously presented) The file replication control method according to claim 23, wherein the node asks another node that shares the file to perform a process for an access request for the file when the access request takes place in the node itself before data is completely restored.

26. (previously presented) The file replication control method according to claim 14, wherein a node that has performed a systematic stop in which nodes that share a file are synchronously stopped to store a systematic stop state and the node synchronously resumes a process for the file without restoring data of the file.

27. (previously presented) A file replication method for a system having a plurality of nodes connected to a network, files being distributed to the nodes, comprising:

causing a first node to request a token for accessing a file;

causing the first node to access the file when the first node has the latest data of the file and is able to obtain the token for accessing the file from another node having update permission for the file which is given to only one node at a time;

notifying the first node of a second node that has the token when the first node is not capable of acquiring the token; and

causing the first node to ask the second node to access the file when the first node is notified that the first node is not capable of acquiring the token.

28. (previously presented) A computer-readable portable storage medium, when being used by a computer that composes a node connected to other nodes through a network in a file replication system, on which is recorded a program for causing the computer to execute a process, said process comprising:

when the node accesses a file and a node itself has the latest data of the file and has or is able to obtain access permission from another node having update permission for the file, causing the node itself to access the file of the node itself; and

when another node has the update permission for the file which is given to only one node at a time, causing the node itself to ask the other node to access the file.